Visualization of metrics and areas of interest on software architecture diagrams

Byelas, Heorhiy

Chapter 9

Conclusion

9.1 Summary

In this thesis, we have studied the creation of visualizations of combined software architecture diagrams and software metrics. For the architectural data, we have used a UML-like representation of the software structure, as UML diagrams are arguably well understood and widely accepted by the main target group of our visualizations, the software architects and designers involved in creating and understanding complex software systems.

Besides the structural data which is inherent to UML diagrams, such as entities and relationships between entities, we also considered a new data type: groups of elements that are related to one concern, or aspect, in a given system design, called *areas of interest*, which is our first research sub–question (see Section 1.4). We have presented methods to render areas of interest on software diagrams in a way that imitates the style actual human users would draw such annotations on paper or whiteboard diagrams.

For metrics, we have presented ways to visualize metrics defined on diagram element members, such as class methods, as well as metrics defined on elements involved in areas of interest, which is our second research sub–question (see Section 1.4)

Overall, the presented techniques work as *annotations* on existing UML-like diagrams, rather than proposing entirely new ways for combining metrics with diagrams, such as, for example, 3D visualizations. Using transparency and blending, all visualized items, such as metrics and areas, can be emphasized, visually pushed in the background, or completely removed from a given UML diagram visualization, without changing the visualized diagram's layout. This enables users to smoothly navigate between a classical, well-understood, diagram view, and a diagram view annotated with additional metric or area-of-interest information.

We have evaluated the proposed visualizations on several levels. First, we have conducted a user study that compared the understandability of our automatically-generated areas of interest with user-drawn areas of interest, and elicited a number of algorithmic improvements that brought our visualizations close to good human drawings. Secondly, we have used our visualizations in several case studies focusing on understanding various

aspects of software architectures, with a focus on reverse engineering and maintainability. Finally, we have used the prototype visualization tool that was constructed during this work in the framework of an academic-industry collaboration, and observed the reactions of actual users with respect to the proposed visualization methods.

9.2 Directions of Future Work

There are several possible directions of future work from the results presented here. In the following, we outline these directions, ordered on their perceived potential to produce useful results for the overall task of helping users understand the correlation of structures with metrics on diagrams.

9.2.1 Metrics on relations

All techniques presented in this thesis consider the visualization of metrics defined on *entities* in the entity-relationship model that underlies a software architecture diagram. However, many such metrics exist also for relationships [52]. For example, for associations that indicate function calls, we can consider the number of times a function is called; the length (duration) of the call; whether the function is virtual, overloaded, static, or a remote procedure call. For associations that indicate data members, we can consider the number of times the member is accessed, and whether the access is a read or a write. For inheritance relations, we can consider the type of inheritance (public, private, or protected); or the amount of interfaces which is defined, specialized, or used within the inheriting class.

It is challenging to consider how to render several such metrics atop of relations in a UML-like diagram, especially if we want to keep our constraints of not modifying a given layout. The problem is not trivial, since typical diagrams can have a large number of relations, whose visual line-like representations can intersect several times. A different way would be to explore visualizations where relations become first-class citizens [72], and see how such methods can be combined with classical UML diagram visualizations.

9.2.2 Relations and areas

Our areas of interest are, so far, strictly defined in terms of entities. However, it may be desirable to explicitly include relations within areas of interest (Section 3.8.2). This would be useful, for example, in the case when certain elements are involved within an area of interest from the perspective of a given relation, and involved in other areas, or no area, from the perspective of another relation. Since a UML-like diagram has a single representation for a given element, we may need ways to make the connection between areas and relations visually explicit on the diagram.

Different routes are possible to address the above goal. First, one could constrain the geometric shapes and positions of areas of interest so that they make explicit which relations between their contained elements these areas include. However, using solely this technique, the visual connection between an area and the relations it includes may be not

sufficient. Secondly, one could design additional visual cues to mark the inclusion of relations within a given area (or areas), such as using specific shading and texturing between that area's contour and the included relations. This technique may give good results, however it can also potentially cause undesired occlusions and visual clutter. Finally, one may consider a way to draw the areas of interest which is radically different from our current Venn-Euler diagram-like shapes, with the aim of making the inclusion of both entities and relations within an area visually explicit.

9.2.3 Visual scalability

As outlined in Chapter 8, there are several limitations to the visual scalability of our visualization methods. Improved results and increased usability can be obtained if we augment the number of method-level and area-level metrics displayed at a given time on a diagram; and if we reduce the visual clutter created by multiple overlapping areas of interest. For the latter goal, it seems possible to adapt or reuse many of the geometric and shape processing methods in existence, such as feature-preserving smoothing or medial axes, so that we construct geometric shapes for the areas which are closer to those quality criteria which are perceived as important by users. Assuming that we have reliably detected which are these quality criteria, and that we can quantify them, the main concern here is to maintain the current robustness and speed of our visualization methods, which are indispensible for their actual use in practice.

9.2.4 Areas of interest that show the system evolution

Our examination of areas of interest has focused on *static* software structures. Both the diagrams, areas of interest, and element and area metrics, are mined from a single version of a software code base. This enables visualizations which arguably bring some insight on that single code base. However, software evolves, as outlined in Chapters 1 and 2. It would be highly interesting to work towards extending the areas of interest and multivariate metrics techniques presented in this thesis in the direction of coping with changing software datasets. This will most surely involve fundamental changes in the visualization design, both at the level of areas of interest and metrics. In particular, it would be interesting to develop methods that are able to target specific evolution questions, like showing the changes in a given area of interest (*e.g.* element appearance or disappearance).

A separate direction of interest is developing methods that help in visualizing the co-evolution of different software systems, at an architectural level. Here, the main question is which visual designs are best suited in showing evolution *similarity* and evolution *difference*.

9.2.5 Different application domains

Essentially, there is little that makes our work here strictly applicable to software engineering diagrams. As outlined in Chapter 2, metrics and areas of interest occur also in datasets emerging from other domains, such as social network and organization analysis. It would be interesting to see whether the techniques presented in this thesis are applicable

Figure 9.1: Areas of interest present in different application domains [87]

to these other domains and datasets, and which modifications may be needed. In particular, it would be challenging to consider relational diagrams defined over 'concrete' spaces, such as geographical maps, and to extend the idea of areas of interest to incorporate the inclusion of entire zones from such maps, rather than the more restricted definition we have now which focuses only on inclusion of nodes from a graph.

As a less technical, but still suggestive, example in this direction of larger applicability of our areas of interest to different domains, we choose to end this thesis with an image taken from a movie [87] which shows a doctor sketching a diagram showing relations between various diseases, an image which is strikingly similar to our own areas of interest on software architecture diagrams.